REMARKS

Claims 1-6 and 8-21 are pending in this application. None of the claims are amended by this Response. Reconsideration of the claims in view of the following remarks is respectfully submitted.

I.   Claim Objections

The Final Office Action objects to claim 8 stating that it is dependent upon a canceled claim. However, claim 8 was amended to correct its dependency to be on claim 1 rather than canceled claim 7 in the Response filed December 18, 2006, to which the Final Office Action states that it is in response. Moreover, in view of the withdrawal of the previous Final Office Action, and the numerous discussions Applicants' representative had with the Examiner, his Supervisor, and Mr. Pinchus Laufer (the SPRE), that the Examiner consulted regarding 35 U.S.C. § 103(c), it is Applicants understanding that the Response filed December 18, 2006 was in fact entered. Thus, claim 8 should stand amended such that its dependency is on claim 1. If this is not the case, then Applicants respectfully request that claim 8 be amended to correct its dependency as set forth in the listing of claims above. Accordingly, Applicants respectfully request withdrawal of the objection to claim 8.

II.   Rejection under 35 U.S.C. § 112

The Final Office Action rejects claims 2-3 and 5 stating that there is insufficient antecedent basis for the phrase "the at least one previously disabled portion" in these claims. Specifically, the Final Office Action states that there is only a first portion that is disabled. Applicants respectfully disagree.

Claim 2 recites "selecting at least one previously disabled portion, and automatically re-enabling the at least one selected previously disabled portion." The "at least one previously disabled portion" recited in claim 2 is not necessarily the "first portion" recited in claim 1, although the first portion could certainly be one of the "at

least one previously disabled portion." The "at least one previously disabled portion" may also be, or at least include, other previously disabled portions that are not the "first portion." Thus, contrary to the assertion made in the Final Office Action, there is sufficient antecedent basis for the features of claims 2-3 and 5. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 2-3 and 5 under 35 U.S.C. § 112, second paragraph.

III.   Rejection Under 35 U.S.C. § 103(a) Based on Chenier and Storisteneau

The Final Office Action rejects claims 1, 6, 9-14, and 16-21 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Chenier (U.S. Patent Application Publication No. 2004/0003383) in view of Storisteneau (CA 2256931). This rejection is respectfully traversed.

Claim 1, which is representative of the other rejected independent claims 9-13 with regard to similarly recited subject matter, reads as follows:

> 1.   A method of editing program code on a data processing system, the program code being suitable for subsequent processing, wherein the method includes the steps of:
> defining at least two portions of the program code;
> selecting a first defined portion of the at least two portions of the program code;
> compressing a representation of the first defined portion in a visual representation of the program code such that content of the first defined portion is not visible in the visual representation of the program code, wherein a second defined portion of the at least two portions of the program code remains visible in the visual representation of the program code; and
> automatically disabling the first defined portion, the disabled first defined portion being excluded from the subsequent processing, wherein the second defined portion is subjected to subsequent processing.
> (emphasis added)

Applicants respectfully submit that neither Chenier nor Storisteneau, either alone or in combination, teach or suggest automatically disabling a first defined portion of program code, the first defined portion being a selected portion whose representation is compressed in a visual representation of the program code such that content of the first defined portion is not visible in the visual representation of the program code. Moreover, Applicants respectfully submit that neither Chenier nor Storisteneau, either alone or in combination, teach or suggest making the alleged combination of teachings from Chenier and Storisteneau or the modifications that would be necessary to arrive at the invention as recited in claim 1 or the other rejected independent claims 9-13. Furthermore, Applicants respectfully submit that even if the alleged combination of Chenier and Storisteneau were possible and one were somehow motivated to make the alleged combination, the combination still would not result in the features of the independent claims 1 and 9-13 being taught or suggested.

Chenier is directed to a system for stripping away unwanted portions of program code. Chenier uses a text file to specify information about the types of portions of program code that are to be stripped away and then processes the program code based on the content of this text file. For example, the text file may specify a particular type of tag to look for in the program code such that those portions of code associated with that type of tag are removed from the program code. Chenier does not teach anything regarding a visual representation of the program code, let alone the compression of a representation of a first portion of code in the visual representation of the program code such that the first portion of code is no longer visible in the visual representation of the program code.

The Final Office Action admits that Chenier does not teach compressing a representation of a first defined portion in a visual representation of the program code. However, the Final Office Action alleges that Storisteneau teaches this feature at page 3, lines 15-17 and in Figures 1-2. Applicants respectfully disagree.

Storisteneau teaches a system for providing a graphical representation of code by first providing a graph having nodes representing components of a program and then allowing a user to select nodes to thereby cause the node to be replaced with a window for editing the associated code portion. Storisteneau is directed to solving the problem of

having to have multiple different representations of a program and the user having to correlate information from these multiple different representations.

With the system of Storisteneau, a graph having nodes and arcs, such as shown in Figure 1, is provided. The user may select a node in the graph, and have it replaced with a window for editing the corresponding source code. The user may then edit the source code within the window. Thus, Storisteneau teaches replacing a node representing a portion of code, with a window in which the actual code may be edited. While displaying the window for editing portions of source code may be considered visually representing a portion of program code and returning from the editing window to the nodal representation of the program may be considered a compression of the source code representation into a nodal representation, these are not the same features as are recited in claim 1, as discussed hereafter.

While Storisteneau teaches a graphical representation of code, nowhere in Storisteneau is there any teaching or suggestion of a need to automatically disable a portion of code whose representation in a visual representation of the code has been compressed. Storisteneau is concerned with visualizing the source code for editing while still allowing a user to see the connections between portions of source code via a nodal graph. Storisteneau is not concerned with disabling portions of code, nor does it provide any suggestion to disable portions of code, let alone doing so automatically for portions of code whose representation in a visual representation of a program have been compressed. Moreover, it would not be obvious to modify Storisteneau to include such a feature of automatically disabling portions of code whose representations have been compressed in a visual representation of program code.

The only element in Storisteneau that could reasonably be interpreted as a compressed representation of a portion of code is the node in the graph representation, see Figures 1-2, of Storisteneau for example. If Storisteneau were to automatically disable code whose representation is compressed in a visual representation of the program code, every portion of code that is represented as a node, i.e. in a compressed state, in the graph visual representation would be disabled, i.e. the entire program would be disabled. This would render the system of Storisteneau inoperable and thus, cannot possibly be a teaching or suggestion in the reference.

While Chenier is directed to stripping out portions of code that are associated with tags specified in a text file, Chenier, likewise does not teach or suggest automatically disabling a portion of code whose representation in a visual representation of program code has been compressed, as has been admitted by the Examiner. Since neither reference alone teaches or suggests this feature, any alleged combination, even if such a combination of the teachings of the references were possible and one were somehow motivated to make the alleged combination, still would not result in the features of the independent claims being taught or suggested.

To the contrary, Applicants respectfully submit that one of ordinary skill in the art would not be motivated to make the alleged combination set forth in the Final Office Action. While both references are directed to editing program code, their teachings of the references are directed to solving completely different problems using completely different solutions and thus, one of ordinary skill in the art would not combine the references in the manner alleged by the Examiner. Chenier is directed to the problem of removing unwanted portions of code and utilizes a text file to identify tags to look for in the code and deleting the portions of code between the tags. Storisteneau is concerned with providing a single representation of code such that portions of code may be edited while the dependencies of code may be visualized through a nodal graph. These are completely different problems from each other, and thus, one of ordinary skill would not even attempt to combine them in the manner alleged by the Examiner.

That is, there is no teaching or suggestion in Chenier regarding any deficiency or desired functionality that the teachings of Storisteneau would satisfy. Similarly, there is no teaching or suggestion in Storisteneau regarding any deficiency or desired functionality that the teachings of Chenier would satisfy. This is because each reference is directed to solving different problems and does not see the need for any mechanism from the other reference in achieving its goal. Thus, there is no teaching or suggestion in the references to make the alleged combination.

The Final Office Action alleges that the motivation is "in order to improve graphical representation of source code and enhance program comprehension" (Storisteneau, page 2, lines 25-27). While this is a nice general goal, it does not address why the specific feature of Storisteneau would be needed or desired in the specific system

of Chenier. Where does Chenier state that there is a need to have a nodal representation of program code such that nodes may be selected and their source code displayed? Chenier is not even concerned with the manner by which the program code is represented to a user but instead is concerned with stripping out unwanted portions of code, i.e. code that is not executed and thus, is unnecessary. Moreover, Storisteneau achieves this goal without the need for any of the mechanisms of Chenier and thus, why would one add the teachings of Chenier to Storisteneau to achieve this purpose?

The only suggestion to even attempt to combine the teachings of Chenier with Storisteneau necessarily comes from a prior knowledge of Applicants' claimed invention and a sole motivation of attempting to recreate the claimed invention having first had benefit of knowing the claimed subject matter. This is impermissible hindsight reconstruction which cannot be used as a proper basis upon which to reject the claims.

Moreover, Applicants respectfully submit that even if the teachings of the references were somehow combinable, and one were somehow motivated to attempt such a combination, *arguendo*, the result still would not be the invention as recited in claim 1 or the other rejected independent claims 9-13. To the contrary, if the teachings of the references were combined, the result would be a system substantially as taught by Chenier where a text file is used to identify tags in code for portions of code that are to be stripped out, but in which the code may be visualized using the graph and edit windows of Storisteneau. There still would be no teaching or suggestion to automatically disable a portion of code that was selected and whose representation in a visual representation of the program code has been compressed. The only teaching regarding such a feature is present in Applicants' claims, not the references.

Thus, for the reasons set forth above, Applicants respectfully submit that neither Chenier nor Storisteneau, either alone or in combination, teach or suggest the features of independent claims 1 and 9-13. Moreover, Applicants respectfully submit that the alleged combination of Chenier and Storisteneau is improper and, even if proper, would not result in the invention as recited in independent claims 1 and 9-13. At least by virtue of their dependency on respective ones of claims 1 and 9-13, the alleged combination of Chenier and Storisteneau does not teach or suggest the features of dependent claims 6,

14, and 16-21. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 1, 6, 9-14, and 16-21 under 35 U.S.C. § 103(a).

In addition to the above, the alleged combination of Chenier and Storisteneau does not teach or suggest the specific features recited in dependent claims 6, 14, and 16-21. For example, with regard to claims 18 and 19, neither reference, either alone or in combination, teaches or suggests that at least one of the at least two portions of the program code has an associated level, and that selecting a first defined portion of the at least two portions of the program code comprises receiving an input specifying a level such that portions of program code equal to or above the specified level are visually represented in the visual representation of the program code. Moreover, the references do not teach or suggest that portions of the program code that are not equal to or above the specified level are automatically compressed in the visual representation of the program code such that they are not visible.

The Final Office Action alleges that these features are taught by Storisteneau at page 4, lines 1-5, page 7, lines 1-5, and in Figures 2-3. Page 3, line 26 to page 4, line 5 and page 7, lines 1-5 read as follows:

> According to a further aspect, the invention provides a method for editing a computer program which consists of the steps of displaying to a user a hierarchical relationship between source code components of the computer program in which each component is represented as a node, and repetitively providing means to allow the user to select one of the nodes for editing and replacing the selected node with an edit window displaying the source code component in order to permit the user to edit the source code component while viewing the hierarchical relationship.

(page 3, line 26 to page 4, line 5)

> ...example:
> i) a list of other components which call on a component associated with a source node 54 which can be opened for editing from a pop-up menu 55;
> ii) an action that opens a new window (not shown) displaying the topology of the current file or project; and
> iii) an action to refresh the Graph View 61, to reflect changes made to the components of the program.

(page 7, lines 1-5)

These sections of Storisteneau merely describe the hierarchical nature of program code which may be displayed by the system of Storisteneau. Similarly, Figures 2-3 of Storisteneau merely show the hierarchical nodal graph representation or program code. However, nothing in these sections, the figures, or any other portion of Storisteneau teach or suggest selecting a first defined portion of at least two portions of the program code by receiving an input specifying a level such that portions of program code equal to or above the specified level are visually represented in the visual representation of the program code. Moreover, nothing in Storisteneau teaches or suggests that portions of the program code that are not equal to or above the specified level are automatically compressed in the visual representation of the program code such that they are not visible. Storisteneau only teaches the replacement of nodes in a hierarchical nodal graph with the corresponding source code edit window.

The other dependent claims recite additional features which, when taken alone or in combination with the features of their respective independent claims, are not taught or suggested by the alleged combination of references. Thus, Applicants respectfully submit that dependent claims 6, 14, and 16-21 are further defined over the alleged combination of references by virtue of the specific features recited in these claims.

IV. **Rejection Under 35 U.S.C. § 103(a) Based on Chenier, Storisteneau, and Endejan**

The Final Office Action rejects claims 2-5, 8 and 15 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Chenier in view of Storisteneau, and further in view of Endejan (U.S. Patent Application Publication No. 2002/0184611). This rejection is respectfully traversed.

Chenier and Storisteneau suffer from the deficiencies discussed above, i.e. neither Chenier nor Storisteneau, either alone or in combination, teach or suggest automatically disabling a portion of code whose representation has been compressed in a visual representation of the program code, as recited in the independent claims. Endejan, likewise, does not teach or suggest such a feature.

Endejan teaches a system in which active and inactive portions of program code are displayed using different fonts and grayscaling. With Endejan, pre-processor directives are used to specify which portions of the program code are active and which are inactive. Based on these pre-processor directives, the active code is displayed in one font and grayscale while the inactive code is displayed in a different font and/or grayscale. The inactive and active code portions may be switched by changing the pre-processor directives (see paragraphs 25-30).

Endejan, like Chenier and Storisteneau, does not teach or even suggest automatically disabling a portion of code whose representation has been compressed in a visual representation of the program code. With the Endejan system, inactive portions of code are displayed with a different grayscale or font. Endejan does not compress the representation of portions of code and then disable the portions of code whose representation has been compressed. To the contrary, Endejan must know ahead of time which portions of code are inactive so that it may properly represent them as having a different font and/or grayscale. Thus, none of the cited references, Chenier, Storisteneau, or Endejan, either alone or in combination, teach or suggest the specific features of claim 1, from which claims 2-5, 8 and 15 depend. Therefore, claims 2-5, 8 and 15 are likewise allowable over the asserted combination of Chenier, Storisteneau, and Endejan for at least the reasons set forth above with regard to claim 1. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 2-5, 8 and 15 under 35 U.S.C. § 103(a).

In addition to the above, neither Chenier, Storisteneau, nor Endejan, either alone or in combination, teach or suggest the specific features of dependent claims 2-5, 8 and 15. For example, with regard to claim 3, the references do not teach or suggest to assign each defined portion of program code to a category of a set including at least one category, the step of selecting the first defined portion and the step of selecting the at least one previously disabled portion including selecting at least one category. The Final Office Action alleges that this feature is taught by Chenier at paragraph 5. Paragraph 5 of Chenier merely discusses the use of a text file to define preprocessor macros and comment flags within the source code that identify portions of source code to be removed through a stripping process. Chenier does not teach or suggest that each portion of program code is assigned a category and that these categories are used to select a portion

of code whose representation in a visual representation of the program code is to be compressed and the corresponding portion of code disabled. Neither does paragraph 5 of Chenier teach or suggest that these categories may be used to select at least one previously disabled portion that is to be re-enabled.
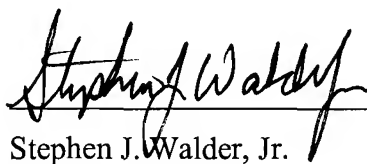
The other dependent claims recite additional features which, when taken alone or in combination with the features of their respective independent claims, are not taught or suggested by the alleged combination of references. Thus, Applicants respectfully submit that dependent claims 2-5, 8 and 15 are further defined over the alleged combination of references by virtue of the specific features recited in these claims.

V.   **Conclusion**

It is respectfully urged that the subject application is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: <u>February 23, 2007</u>

Stephen J. Walder, Jr.
Reg. No. 41,534
Walder Intellectual Property Law, P.C.
P.O. Box 832745
Richardson, TX 75083
(214) 722-6419
ATTORNEY FOR APPLICANTS